PCT/AU2004/000469

Australian Government

Patent Office
Canberra

I, LEANNE MYNOTT, MANAGER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. 2003901714 for a patent by CHARISMATEK SOFTWARE METRICS as filed on 10 April 2003.

WITNESS my hand this
Twenty-eighth day of April 2004

LEANNE MYNOTT
MANAGER EXAMINATION SUPPORT AND SALES

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Invention Title: **AUTOMATIC SIZING OF SOFTWARE FUNCTIONALITY**

The invention is described in the following statement:

AUTOMATIC SIZING OF SOFTWARE FUNCTIONALITY

FIELD OF THE INVENTION

The invention relates generally to software measurement and estimation, and in particular to a method and apparatus for assessing the Functional Size of a software application based upon the software requirements.

BACKGROUND OF THE INVENTION

Within the information technology (IT) industry, it is important to be able to estimate parameters such as the projected development time and cost required to complete a software project, for the purposes of project management. In order to make such estimates, it has become common in the art to first establish a quantity usually referred to as the "Functional Size" of the software, based upon software requirements typically defined in business terminology.

The Functional Size is usually independent of the development environment and technology used to build a software application. Extensive research and practical application have demonstrated that the Functional Size has a direct and consistent relationship to the effort required to design, build and support software applications.

Once the Functional Size has been assessed, the results may be used as the basis for the extrapolation of parameters such as the time, effort or cost associated with development. The most common form of extrapolation is the simple and direct multiplication of the Functional Size by an anticipated overall productivity rate, which may be dependent upon the technology used and other factors present in the development environment.

The most widely accepted method for assessing the Functional Size of a software application is Function Point Analysis. Function Point Analysis evaluates the application's capabilities from the point of view of the end user. It is thus possible to use a formal Requirements Specification of the application as the starting point for Function Point Analysis, however an analysis can be performed based on other types of documentation describing the application.

Function Point Analysis involves assessing the number of "Function Points" corresponding to the required functionality of the software application.

This requires the elements of functionality required within the application to be identified and classified, such that a number of Function Points can be associated with each element. There are several variants of this process, however the most common is the International Function Point Users Group (IFPUG) Counting Practices method. This method is normally performed by experts who are skilled and experienced with respect to the application of this technique. Accordingly, at present, effective implementation of software sizing using Function Point Analysis requires the involvement of people with sufficient up-to-date expertise and experience. The sizing and estimation process is therefore generally perceived to be an additional burden and overhead on the software development process that involves additional time and expense.

Accordingly there exists a need to improve the process of assessing Functional Size in order to reduce or eliminate the need for expert practitioners. Prior art methods for Functional Size estimation include: sizing from code; generation from Computer Aided Software Engineering (CASE) tools; expert system tools; and software tools with advanced user interfaces.

The method of sizing from code is based on using software tools to analyse program code in an attempt to identify functionality from indications of transactions that are present in the code. However, this process is fundamentally flawed in concept and impractical in practice. Any actual code is highly dependent for its form upon the style and technique of an individual programmer. The same functionality may be implemented by different programmers in dramatically different ways, making it impossible to create an analysis tool that is able to produce consistently reliable estimates of Functional Size from code. The required relationship between functional sizing artefacts and code artefacts simply does not exist. Studies carried out by the David Consulting Group in the USA indicate size estimates using this method may be around 500% of actual size. Furthermore, although there are some uses for Functional Size estimates generated after the application has been built, such as estimating ongoing support and maintenance costs, an estimate generated from code is usually too late for the vital purpose of projecting development cost and time.

The concept of generating size estimates from CASE tools is intended to improve upon the method of sizing from code by using data held within a design

automation tool. This data is generally available earlier in the development process, and is less idiosyncratic than program code. Even so, the logical view of the software which is required for good functional sizing is quickly lost in the artefacts generated by CASE tools. Consequently, attempts to improve size estimation in this way have suffered from many of the same problems in accuracy experienced with sizing from code. Studies carried out by the David Consulting Group in the USA indicate size estimates using this method are often around 300% of actual size.

Since effective functional sizing is most commonly performed by experts applying rules such as those of the IFPUG Counting Practices Rules Committee, another approach to improving the process has been to build these rules directly into Expert System style tools. However, the complexity of the rules, and the problems of translating the language in which they are expressed into the form required for implementation has led to results that are largely unworkable, and impractical for most serious applications. Consequently, they do not meet the need to significantly reduce dependence upon human experts, and they have failed to solve the problems of time and expense associated with estimating Functional Size.

In another attempt to solve these problems, the work of functional sizing experts is "streamlined" through the use of software tools providing an advanced user interface for the entry and management of functional information. At their most effective, such tools provide a "point and click" approach to classification, and useful defaults for complexity of functions, enabling the operator to perform an analysis more quickly, efficiently and consistently than would be possible by hand, or using less sophisticated tools. However, even the most advanced tools do not integrate completely with existing practices and CASE tools, and there is still a need for substantial expertise on the part of the user. Thus provision of an advanced user interface is only a partial solution to the problem, which does not enable the process to be performed without the assistance of skilled personnel.

It is accordingly an object of the present invention to provide a method of assessing the Functional Size of a software application that mitigates one or more of the disadvantages in the prior art.

Any discussion of documents, devices, acts or knowledge in this specification is included to explain the context of the invention. It should not be taken as an admission that any of the material formed part of the prior art base or the common general knowledge in the relevant art on or before the priority date of the claims herein.

SUMMARY OF THE INVENTION

According to one aspect, the present invention provides a method for assessing a Functional Size of a software application or project including the steps of:

analysing a software Requirements Specification and determining zero or more keywords for each requirement;

cross-referencing the keywords with a lexicon and associating each keyword with an entry in the lexicon;

obtaining a function type and complexity for an associated lexicon entry for each keyword;

using a functional sizing standard to deduce a number of function points associated with the function type and complexity for each relevant lexicon entry; and

combining the function points to obtain a Functional Size of the application or project.

In a preferred embodiment, the step of analysing the Requirements Specification includes parsing each requirement to isolate lexical elements, and determining a keyword corresponding to each lexical element. The step of cross-referencing the keywords with the lexicon may include searching the lexicon for entry which is an exact match for each keyword. Alternatively or additionally, the step of cross-referencing may include searching the lexicon for entries which are near matches or equivalents for each keyword.

In a particularly preferred embodiment, the sizing standard is the standard maintained by the International Function Point Users Group. Preferably, the function types that are associated with each keyword in the lexicon include: Internal Logical Files; External Interface Files; External Inputs; External Outputs; and External Enquiries. The complexity that is associated with each keyword

may include Low, Average and High Complexity. Of course, many alternative complexity scalings may be used.

The step of combining the function points may include summing the function points associated with all keywords identified to obtain the Functional Size as the total number of function points.

Preferably, the method also includes the step of deducing at least one parameter associated with management of the development of the software application. The at least one parameter may include one or more of: cost; effort; and development time. In one embodiment, the step of deducing the at least one parameter includes multiplying the Functional Size by a number representing a corresponding productivity rate, however more sophisticated calculations may be used that include factors such as the variations of productivity in different environments.

According to another aspect, the present invention provides a system for assessing a Functional Size of a software application or project including:

a lexicon for storing lexicon keywords and function type and complexity data associated with the lexicon keywords;

input means for entering a software Requirements Specification;

processing means for processing the software Requirements Specification, said processing including the steps of:

analysing the software Requirements Specification to associate each requirement with zero or more requirement keywords;

cross-referencing the requirement keywords with the lexicon to obtain a corresponding function type and complexity from the lexicon by matching requirement keywords with lexicon keywords;

computing a number of function points associated with each function type and complexity using the rules of a sizing standard; and

combining the function points to obtain a Functional Size of the application or project.

Preferably, the lexicon is able to be modified and extended by a user, so that the lexicon keywords and corresponding function type and complexity can be changed, and new lexicon keywords with corresponding function type and complexity can be added.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the invention will now be described, without limiting the overall scope of the invention, with reference to the accompanying drawings in which:

Figure 1 is a logical data flow diagram illustrating an exemplary system for assessing the Functional Size of a software application or project.

DESCRIPTION OF PREFERRED EMBODIMENT

Figure 1 illustrates an exemplary system 100 for assessing the Functional Size of a software application in the form of a logical data flow diagram. In the exemplary embodiment, the main source of information to be analysed and processed by the system is a Requirements Specification document 102 containing the base requirements of the user. The Requirements Specification defines a list of statements defining the functional features that are to be implemented in the software application.

Each requirement in the specification is input to a process 104 that parses the requirement to identify possible keywords that are associated with the requirement. In one embodiment of the invention, the parsing process 104 extracts individual words or phrases from the requirement. However in other embodiments the parsing process 104 may perform more sophisticated analysis of the text of the requirement in order to infer possible keywords. The parsing process 104 may also be able to identify portions of the requirement text that perform other functions, such as identifying the objects or elements within the software application to which the functional requirement refers.

The extracted list of possible keywords is passed to a keyword look-up process 106, the function of which is to search a lexicon file or database 108 for any entries corresponding to the keywords. In the preferred embodiment, the keywords in the lexicon are ordinary words chosen on the basis of their probable association with specific functional elements defined by the Requirements Specification. If no corresponding keywords are found, then the requirement is assumed to consist of a linking component that does not specify any functional element with which a size is associated. In this case, no further processing is performed, and the system will continue processing the next requirement.

If any keyword is found in the lexicon 108, it is passed to a process 110, the function of which is to determine a classification for the functional element of the software application represented by the keyword. In the preferred embodiment, the classification consists of a function type and a complexity, which are stored in the lexicon 108 along with the corresponding keyword. More particularly, in the common method of Function Point Analysis, the function type is either a Data Function or a Transactional Function. Data Functions are either Internal Logical Files or External Interface Files, while Transaction Functions are one of External Inputs, External Outputs, or External Inquiries. The complexity is then one of Low, Average or High. Thus an example of a valid classification is "External Inquiry of Average Complexity."

Once the classification of the functional element has been determined, it is passed to a process 112, the purpose of which is to deduce the Function Point Value of the functional element based on its classification. In the preferred embodiment, the Function Point Value is obtained by the use of a Sizing Standard, such as the International Function Point Users Group Guidelines. The required data corresponding to the chosen standard may be stored in a file or database 114 for look up by the process 112.

The Function Point Values are input to a function point accumulation process 116. In the presently preferred embodiment, this process calculates the sum of all the input Function Point Values, however it will be appreciated that other methods for accumulating the Function Point Values are possible.

The output of the process 116 is an overall Functional Size for the software application, which may be input to a parameter extrapolation process 118. The function of the extrapolation process 118 is to compute one or more parameters associated with management of the development of the software application or project. The parameters that could be computed include cost, development effort or development time. In one embodiment, any of these parameters may be obtained by multiplying the Functional Size by a number representing a corresponding productivity rate. It will be appreciated by those skilled in the art that more sophisticated calculations may be used that include factors such as the variations of productivity in different environments.

In a particularly preferred embodiment, the system is implemented in a computer program designed to run on a general purpose computer. Specifically, the program may be designed for use on a personal computer running a version of the Microsoft Windows operating system. Consequently, a number of methods may be provided for the user to enter a Requirements Specification for processing by the program. For example, the user may be able to manually enter each requirement for processing. Since the Requirements Specification is generally a separate document generated as part of the overall software development program, it is usually more convenient to enable the user to enter a complete set of requirements that are created and maintained using another software application, such as a word processing program, spreadsheet program, or requirements definition program. In this regard, suitable commercially available source applications include Word or Excel from Microsoft, and DOORS from Telelogic. The method of entry of the requirements may be via import of the requirements document from a file generated by the source application, or simply by using the standard cut-and-paste facility via the Windows clipboard. It will of course be appreciated by those skilled in the art that the system could equally be implemented as a program for use on other hardware and operating system platforms, or to use other methods for entering requirements for processing.

Ideally, the lexicon used by the program is a file or database that is itself accessible and extendible to the end user. For this purpose, a software tool may be provided for viewing and editing the contents of the lexicon, either as a separate application or as a part of the program implementing the automated sizing system. This tool enables the user to view the keywords currently contained in the lexicon, along with the classifications associated with each keyword. It also enables the user to delete or replace keywords, to change the classifications associated with the keywords, and to enter new keywords and associated classifications into the lexicon. Entry of new keywords and classifications may be manual, for example via a dialog box entry system, or it may be via a bulk import facility in which the new keywords and classifications are inserted from a file or using cut-and-paste via the clipboard from another application such as a word processor or spreadsheet program.

In order to further clarify the operation of the preferred embodiment of the system, a simple example is presented. For the purpose of this example, a simple lexicon is assumed to contain the following entries:

Keyword: "add" **Classification**: External Input of Average Complexity

Keyword: "delete" **Classification**: External Input of Average Complexity

Keyword: "update" **Classification**: External Input of Average Complexity

Keyword: "search" **Classification**: External Inquiry of High Complexity

The customer requirements have been defined within a Requirements Specification document prepared using Microsoft Word, and the document has been automatically indexed using the standard Word Table of Contents feature. In this example, processing of the following section of the Requirements Specification by the automated sizing system is described:

3.17.2.2 Online – Location Maintenance Menu

3.17.2.3 Online – Locn Maint – Add

3.17.2.4 Online – Locn Maint – Search

3.17.2.5 Online – Locn Maint – Update

These requirements are entered into the automated sizing program by cutting and pasting the above lines from the Table of Contents of the Word document.

Each line is then parsed by the program to identify possible keywords, which are then compared with the keywords stored in the lexicon. When a keyword is found in the lexicon, the program extracts the associated classification information, and creates a transaction of the corresponding type and complexity.

Consequently, after parsing this example, the first line, with number 3.17.2.2, is found to contain no keywords currently stored in the lexicon, and is simply held as a linking component with no associated Functional Size.

The remaining lines in the example each contain a matching keyword within the lexicon. The remaining text is interpreted by the program as identifying the object of the keyword. Thus the lines numbered 3.17.2.3, 3.17.2.4 and 3.17.2.5 indicate that the functions Add, Search and Update are to be provided acting upon an object called "Locn Maint" (Location Maintenance). From the lexicon, it is found that the Add and Update functions are both External Inputs of

Average Complexity, and the Search function is an External Inquiry of High Complexity.

Establishment of the Functional Size of these transactions is completed through the use of a sizing standard. Assuming the International Function Point Users Group Guidelines are applied, an External Input of Average Complexity is allocated the relative Function Point Value of 4, whereas an External Inquiry of High Complexity is allocated the relative Function Point Value of 6.

The Functional Size of the section of the Requirements Specification shown in this example can be calculated by adding the individual Function Point Values corresponding to each requirement, which are 0, 4, 6 and 4 respectively. The Functional Size is thus 14 Function Points.

The Functional Size may then be used to estimate specific parameters of the software development process. For example, if an estimate of the development effort in person hours is required, a simple extrapolation based upon multiplication of the Functional Size by a suitable factor may be used. Accordingly, an estimate of around 50-60 person hours effort may be deduced assuming a simple environment, although it will be appreciated by those skilled in the art that more sophisticated extrapolations are possible using additional data to refine probable delivery rate productivity.

CONCLUSION

A computer program embodying the method of the present invention may be employed to assess the Functional Size of a software application based upon a Requirements Specification document produced as part of the usual process of software development. Accordingly, the problems associated with performing the size assessment as a separate task incurring additional time and expense may be mitigated. Furthermore, the dependence of the process upon experts with specialised skills, experience and knowledge is reduced. The ability, in a preferred embodiment, to modify and extend the lexicon used to identify and classify functional elements ensures that the system is sufficiently flexible to support the needs of a wide range of users and to enable improvements over time.

The present invention is not limited in scope to the described embodiment or embodiments but rather the scope of the present invention is broader so as to encompass other forms of the system, other methods of entering and parsing requirements and identifying corresponding keywords, alternative sizing standards, and other methods of extrapolating Functional Size to obtain estimates of software development parameters.

DATED this 10th day of April 2003

CHARISMATEK SOFTWARE METRICS PTY LTD

WATERMARK PATENT & TRADE MARK ATTORNEYS
290 BURWOOD ROAD
HAWTHORN VICTORIA 3122
AUSTRALIA

P22638AUP1 V2 : NWM/MAS/RES

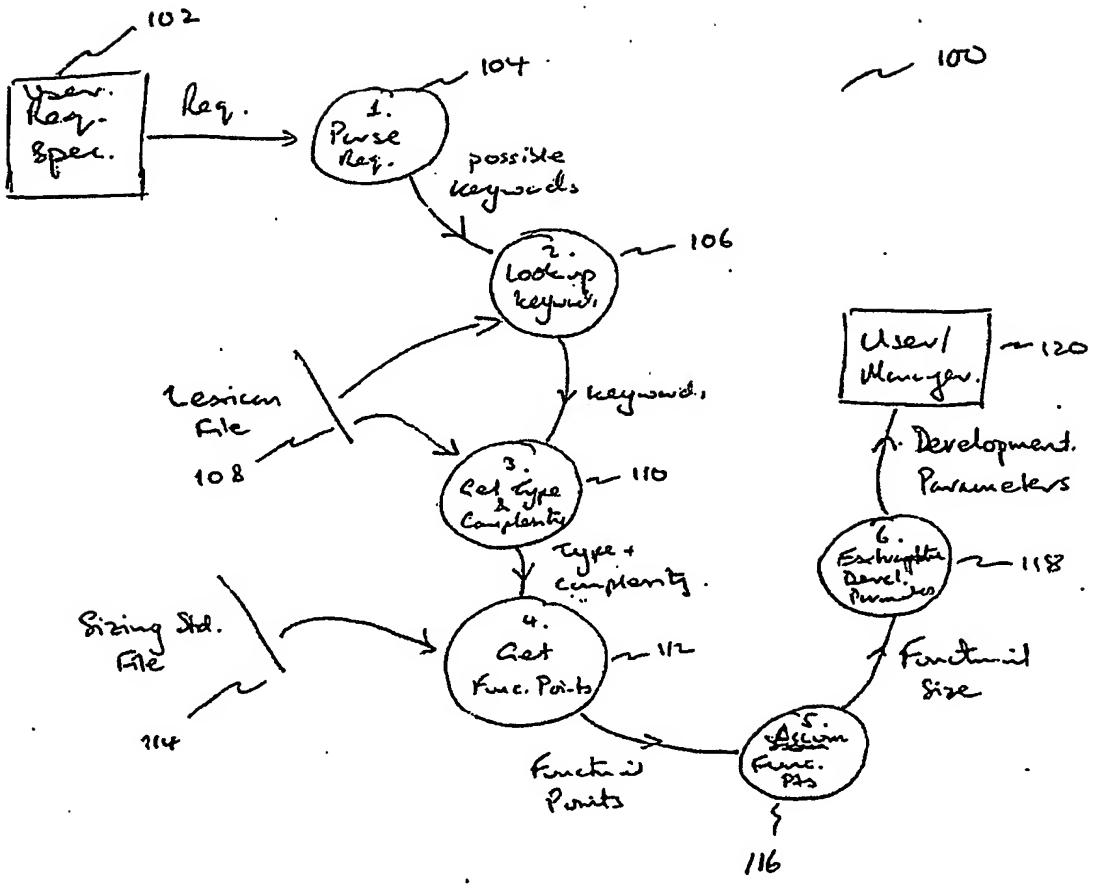
102
User Req. Spec.
Req.
104
1. Parse Req.
possible keywords
100
2. Look up keywords
106
User/ Manager
120
Lexicon File
108
keywords
3. Get Type & Complexity
110
Development Parameters
6. Establish Devel. Parameters
118
Type + complexity
Sizing Std. File
114
4. Get Func. Pts
112
Function Size
5. Assign Func. Pts
Function Points
116

FIGURE 1

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY

(Chapter II of the Patent Cooperation Treaty)

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference P22638PCAU	**FOR FURTHER ACTION**	See Form PCT/IPEA/416
International application No. **PCT/AU2004/000469**	International filing date *(day/month/year)* 8 April 2004	Priority date *(day/month/year)* 10 April 2003

International Patent Classification (IPC) or national classification and IPC

Int. Cl. [7] G06F 09/44

Applicant

CHARISMATEK SOFTWARE METRICS PTY LTD et al

1. This report is the international preliminary examination report, established by this International Preliminary Examining Authority under Article 35 and transmitted to the applicant according to Article 36.
2. This REPORT consists of a total of 3 sheets, including this cover sheet.
3. This report is also accompanied by ANNEXES, comprising:
   a. ☐ *(sent to the applicant and to the International Bureau)* a total of sheets, as follows:
      ☐ sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications authorized by this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions).
      ☐ sheets which supersede earlier sheets, but which this Authority considers contain an amendment that goes beyond the disclosure in the international application as filed, as indicated in item 4 of Box No. I and the Supplemental Box.
   b. ☐ *(sent to the International Bureau only)* a total of (indicate type and number of electronic carrier(s)) , containing a sequence listing and/or table related thereto, in computer readable form only, as indicated in the Supplemental Box Relating to Sequence Listing (see Section 802 of the Administrative Instructions).
4. This report contains indications relating to the following items:
   - ☒ Box No. I Basis of the report
   - ☐ Box No. II Priority
   - ☐ Box No. III Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
   - ☐ Box No. IV Lack of unity of invention
   - ☒ Box No. V Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
   - ☐ Box No. VI Certain documents cited
   - ☐ Box No. VII Certain defects in the international application
   - ☐ Box No. VIII Certain observations on the international application

Date of submission of the demand	Date of completion of the report
8 November 2004	4 March 2005
Name and mailing address of the IPEA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized Officer **John Thomson** Telephone No. (02) 6283 2214

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY

International application No.
PCT/AU2004/000469

Box No. I Basis of the report

1. With regard to the **language**, this report is based on the international application in the language in which it was filed, unless otherwise indicated under this item.

☐ This report is based on translations from the original language into the following language which is the language of a translation furnished for the purposes of:

☐ international search (under Rules 12.3 and 23.1 (b))

☐ publication of the international application (under Rule 12.4)

☐ international preliminary examination (under Rules 55.2 and/or 55.3)

2. With regard to the **elements** of the international application, this report is based on (*replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report*):

☒ the international application as originally filed/furnished

☐ the description:

pages as originally filed/furnished
pages* received by this Authority on with the letter of
pages* received by this Authority on with the letter of

☐ the claims:

pages as originally filed/furnished
pages* as amended (together with any statement) under Article 19
pages* received by this Authority on with the letter of
pages* received by this Authority on with the letter of

☐ the drawings:

pages as originally filed/furnished
pages* received by this Authority on with the letter of
pages* received by this Authority on with the letter of

☐ a sequence listing and/or any related table(s) - see Supplemental Box Relating to Sequence Listing.

3. ☐ The amendments have resulted in the cancellation of:

☐ the description, pages
☐ the claims, Nos.
☐ the drawings, sheets/figs
☐ the sequence listing (*specify*):
☐ any table(s) related to the sequence listing (*specify*):

4. ☐ This report has been established as if (some of) the amendments annexed to this report and listed below had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).

☐ the description, pages
☐ the claims, Nos.
☐ the drawings, sheets/figs
☐ the sequence listing (*specify*):
☐ any table(s) related to the sequence listing (*specify*):

* *If item 4 applies, some or all of those sheets may be marked "superseded."*

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY

International application No.
PCT/AU2004/000469

Box No. V Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Claims **1-28**	**YES**
	Claims	**NO**
Inventive step (IS)	Claims **1-28**	**YES**
	Claims	**NO**
Industrial applicability (IA)	Claims **1-28**	**YES**
	Claims	**NO**

2. Citations and explanations (Rule 70.7)

Claims 1-28 The closest prior art was found to be:

- US 2003/0033586 A
- GB 2362483 A
- US 6073107 A
- US 2003/0070157 A

No individual citation nor obvious combination of citations discloses a method for assessing a functional size of a software application including the steps of analysing a software requirements specification and determining zero or more keywords for each requirement, using a computer to cross-reference the keywords with a stored lexicon including a function type and complexity for each keyword, and further using the computer to associate each keyword with an entry in the lexicon thus obtaining a function type and complexity for each keyword, using a functional sizing standard to deduce a number of function points for the function type and complexity of each keyword, and combining the function points to obtain a functional size of the software application.